JavaScript

About

A light discussion of JavaScript in FreeSWITCH™.

V8 Supersedes SpiderMonkey

FreeSWITCH originally used the Mozilla SpiderMonkey JavaScript (ECMAScript) engine. It supports all the standard JavaScript language elements, for example 'for' and 'while' loops, regexps, etc. In addition there are things that are specific to FreeSWITCH which are listed below. You need to have mod_v8 compiled and installed to take advantage of these functions. Each of those sections is documented below.

From commit b76233266931ca0b53f3bcc589277b972d714216 (January 14'th 2014) in git master, FreeSWITCH has support for the Google V8 JavaScript (ECMAScript) engine. This is provided from the module mod_v8. In current git master mod_v8 is the default JavaScript engine. mod_v8 is a drop-in replacement for mod_spidermonkey, so the old scripts should work as before. There are two differences though, the SpiderMonkey engine has support for built in classes XML and File; these classes are not available in the V8 engine. mod_v8 provides a different XML interface, you can read more about it here: Javascript XML. mod_v8 will publish similar functions as the File class, but this is still work in progress so check the status of that functionality.

Build and Install

mod_v8 is now built and installed by default.

If for some reason it's not, do the following:

- Uncomment languages/mod_v8 in modules.conf in your src directory (make it always build and install v8)
- Run 'make mod_v8-install' to make and install just the v8 module
- Edit conf/autoload_configs/modules.conf.xml in your FreeSWITCH™ install directory to load mod_v8
- Restart freeswitch

Execution of a script

There are two methods for script execution, via an API call *jsrun* and via an application *javascript*. When you use the API a default Session object is not created, as there is no call to associate it with. By default, scripts will be loaded from the {FS_ROOT}/scripts directory, you may override this by placing a / as the first character in Unix-like systems or a x:\ in Windows systems as the script name.

Application

From the Dialplan, you simply call it as an application similar to:

```
<action application="javascript" data="/path/to/some/script.js"/>
```

If you need to pass arguments into to the javascript application, they can be passed in as a single string to the variable "argv" by doing the following:

```
<action application="javascript" data="/path/to/some/script.js $1"/>
```

API

At the FreeSWITCH™ console, or some other application/interface execute the jsrun call with the script name as its argument.

```
jsrun /path/to/some/script.js
```

Examples

- Javascript Hello World
  - A quick-start guide through creating and launching your first Javascript application.
- Javascript Examples exist and include database connectivity and grabbing external content via CURL.

Function Reference

- Misc Javascript functions

SpiderMonkey Specific Objects

- File - File IO methods (Spidermonkey object)

Other Points Of Interest

- Run - Use CURL from within your Spidermonkey script
- Javascript_Event
- Javascript_XML

FAQ

Passing variables to my JavaScript

On console:

```
jsrun script.js arg1 arg2 ...
```

These can be accessed as argv[0], argv[1],.. in the script

In dialplan

Dialplan JS example

```
<extension ...>
  <condition ...>
     <action application="javascript" data="script.js ${caller_id_number}"/>
   </condition>
 </extension>
```

If you're running the script as an application for originate command:

JS originate examples

```
//single argument
   originate sofia/internal/1000 &javascript(script.js arg1)

//multiple arguments
   originate sofia/internal/1000 '&javascript(script.js arg1 arg2)'
```

How can I print stuff inside my JavaScript

```
console_log("notice","<Whatever you need to print>");
```

See Also

- Mozilla JavaScript reference